

# An Overview of the Interoperability Roadmap for COM/.NET-Based CAPE-OPEN

William M. Barrett(a), Michel Pons(b), Lars von Wedel(c), and Bertrand Braunschweig(d)

*a) US Environmental Protection Agency, Cincinnati*

*b) CO-LaN, Rueil-Malmaison*

*c) AixCAPE e.V. , AACHEN*

*d) IFP, Rueil Malmaison*



# Summary

- ◆ The CAPE-OPEN standards were created to allow process modelling components (PMCs) to be used in any process modelling environment (PME) utilizing these standards.
- ◆ The CAPE-OPEN specifications were based upon both Microsoft's Component Object Model (COM) and the Object Management Group's (OMG) Common Object Broker Architecture (CORBA).
- ◆ Since the inception of the CAPE-OPEN project, Microsoft updated COM to the .NET Framework.
- ◆ CO-LaN provides a roadmap for the evolution of CAPE-OPEN into the .NET environment.

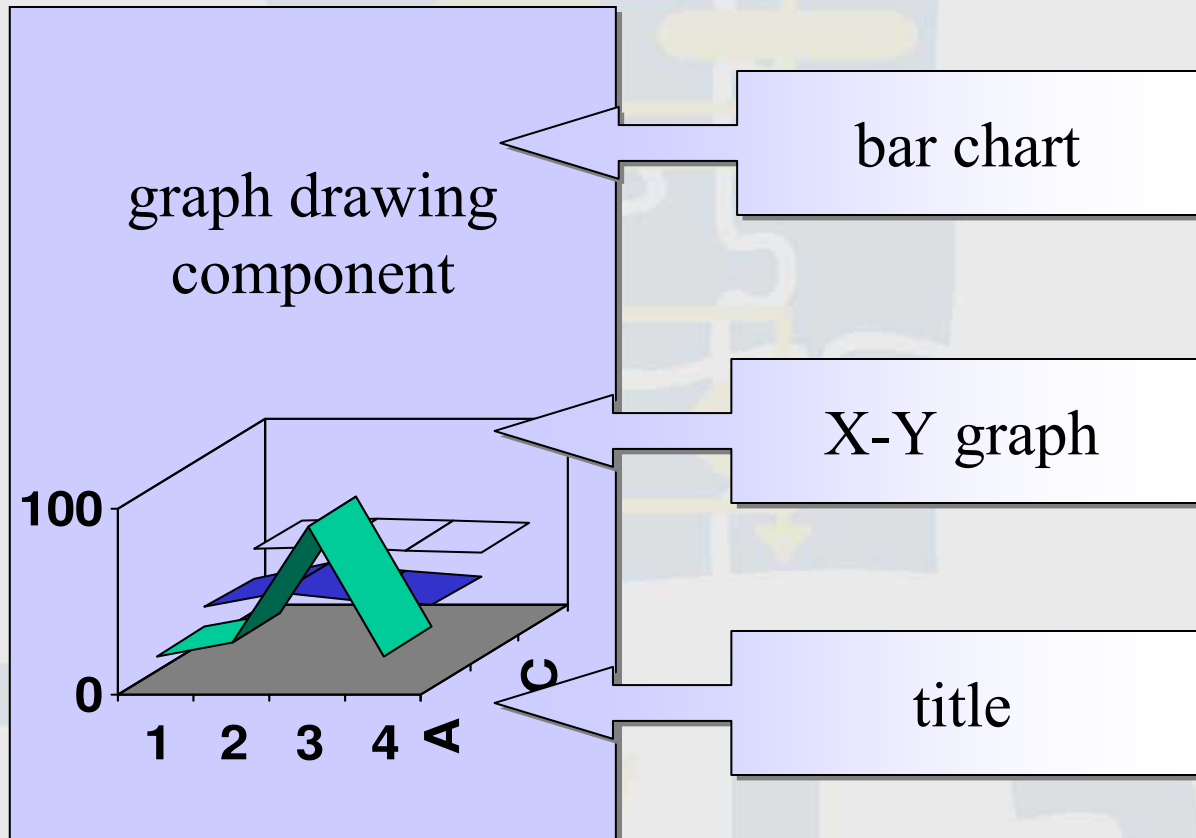


# Outline

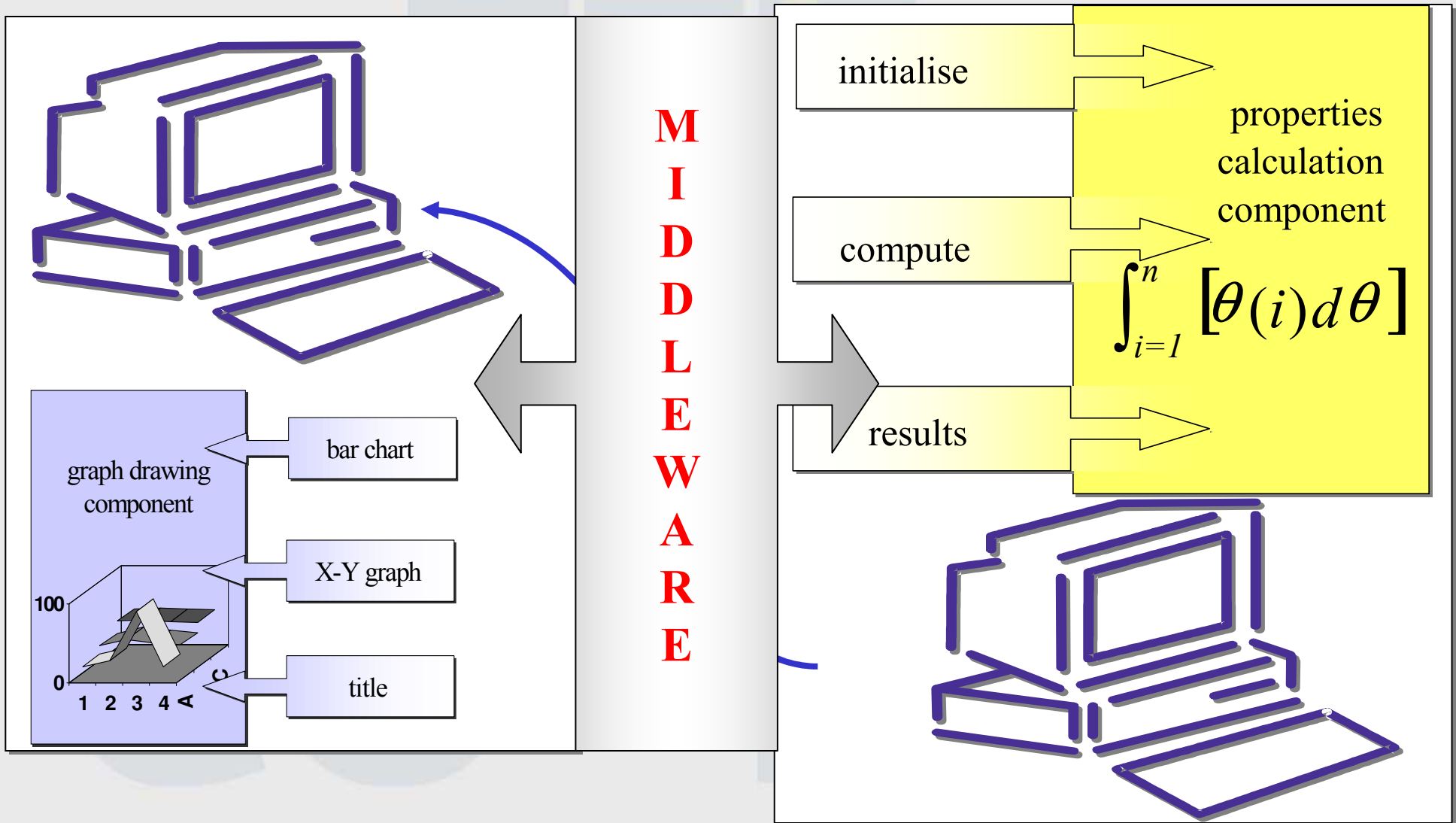
- ◆ **The role of middleware in CAPE-OPEN**
- ◆ **Why .NET, and what it is**
- ◆ **.NET/COM CAPE-OPEN Interoperability**
- ◆ **CO-LaN's roadmap for .NET**



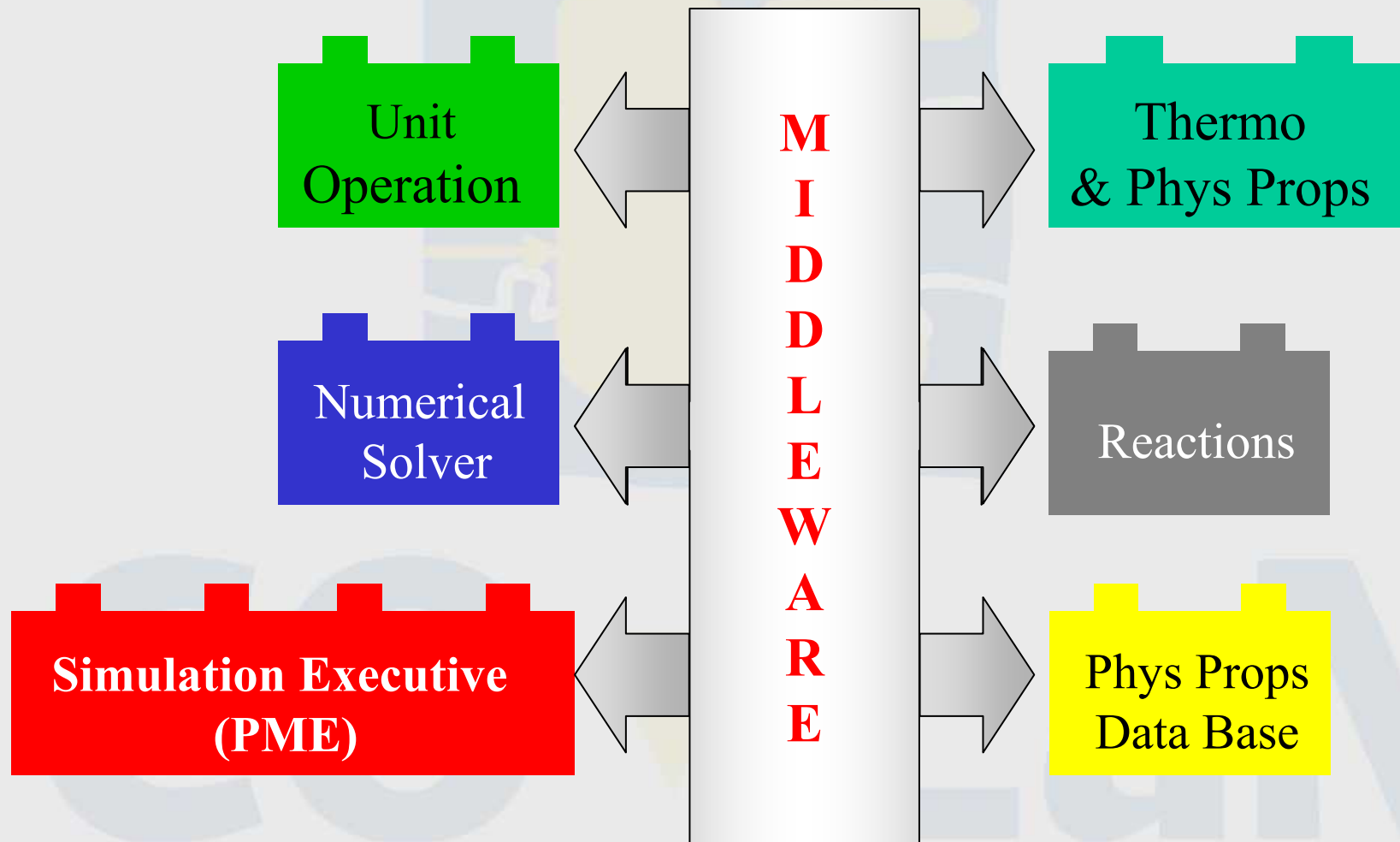
# software components



# distributed software components



# CAPE-OPEN Components



# Models & Middleware for distributed components

- ◆ **Microsoft's COM/DCOM**
- ◆ **OMG's CORBA component model**
- ◆ **Sun' Java-based J2EE (EJB)**
- ◆ **Web Services and Service-Oriented Architectures (SOAs)**
- ◆ **FIPA's multi-agent standards**
- ◆ **Microsoft .NET**
- ◆ **etc..**



# CAPE-OPEN choice until now

- ◆ **Microsoft's COM/DCOM**
- ◆ **OMG's CORBA** component model
- ◆ **Sun' Java-based J2EE (EJB)**
- ◆ **Web Services and Service-Oriented Architectures (SOAs)**
- ◆ **FIPA's multi-agent standards**
- ◆ **Microsoft .NET**
- ◆ **etc..**



# What Is COM?

- ◆ **COM is a standard that provides for:**
  - ⇒ Language independence.
  - ⇒ Location independence.
- ◆ **COM Interfaces**
  - ⇒ Objects implement multiple interfaces
  - ⇒ Clients can select a specific interface to use
- ◆ **In addition**
  - ⇒ It guarantees a common behaviour for all objects through the interface IUnknown.
  - ⇒ It handles the components management processes (instanciation, reference counting,... )
  - ⇒ It provides standard error/exception handling.

# COM Interface

- ◆ **Contract between objects**
  - ⇒ Defines group of functions supported
  - ⇒ Defined in a language called IDL
- ◆ **Strongly typed**
  - ⇒ Each interface is unique and has a unique IID
  - ⇒ Confusion between interfaces cannot happen accidentally
  - ⇒ Defines binary standard through which objects communicate
- ◆ **Robust under change.**
  - ⇒ Objects can support additional interfaces over time without breaking existing clients

# IDL definitions are stored in a Type Library

The screenshot shows the ITypeLib Viewer application window. The title bar reads "ITypeLib Viewer". The menu bar includes "File" and "View". The toolbar contains icons for saving, opening, and help. The left pane displays a tree view of the interface hierarchy:

- interface ICapeUnit
  - m ports
  - m parameters
  - m dirty
  - m valid
  - m Calculate
  - m Restore
  - m Save
  - m Initialize
  - m Terminate
  - m Validate
  - m simulationContext
  - Inherited Interfaces
    - IDispatch
      - m GetTypeInfoCount
      - m GetTypeInfo
      - m GetIDsOfNames
      - m Invoke
    - Inherited Interfaces
      - IUnknown

The right pane displays the IDL code for the ICapeUnit interface:

```
[  
    odl,  
    uuid(678C0998-7D66-11D2-A67D-  
00105A42887F),  
    helpstring("ICapeUnit Interface"),  
    dual,  
    oleautomation  
]  
interface ICapeUnit : IDispatch {  
    [id(0x00000001), propget,  
helpstring("Gets the whole list of  
ports")]  
    HRESULT ports([out, retval]  
IDispatch** ports);  
    [id(0x00000002), propget,  
helpstring("Gets the whole list of  
parameters")]  
    HRESULT parameters([out, retval]  
IDispatch** parameters);  
    [id(0x00000003), propget,  
helpstring("Informs of the persistent  
state of the unit")]  
    HRESULT dirty([out, retval]  
VARIANT_BOOL* isDirty);  
    [id(0x00000004), propget,  
helpstring("Informs of the validation
```

## Situation with COM

- ◆ **90% of CAPE-OPEN applications have been using COM as middleware**
  - ⇒ **CORBA used in a few cases**
- ◆ **Both are considered to be highly successful**
- ◆ **But**
  - ⇒ **Now .NET is here and MS will phase out COM**
  - ⇒ **« Microsoft recommends that developers use the .NET Framework rather than COM for new development »**

# Outline

- ◆ The role of middleware in CAPE-OPEN
- ◆ **Why .NET, and what it is**
- ◆ .NET/COM CAPE-OPEN Interoperability
- ◆ CO-LaN's roadmap for .NET



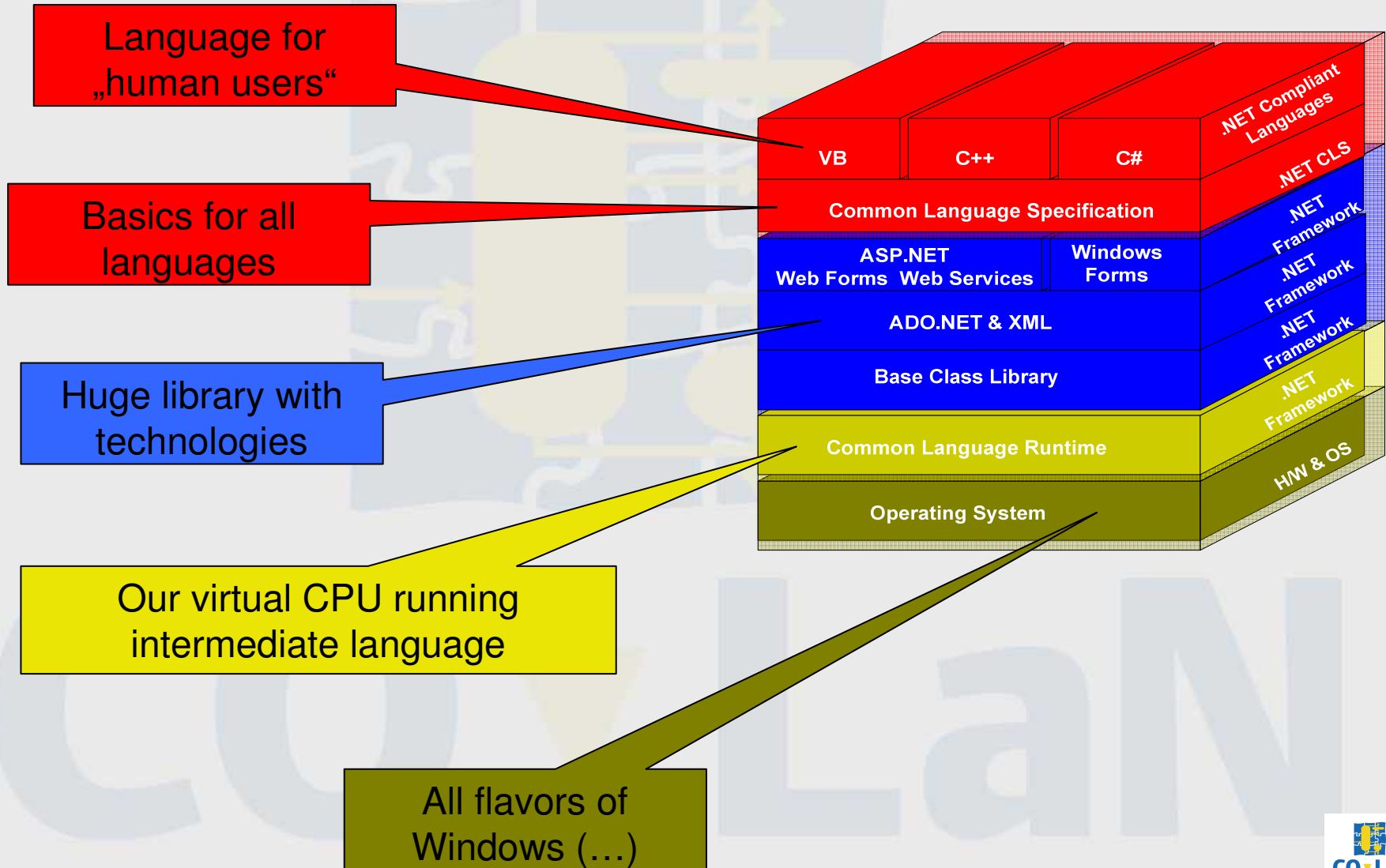
# Why .NET?

- ◆ **The Microsoft .NET framework was created around the late 1990s by Microsoft with several goals in mind:**
  - ⇒ **Unification of the various development technologies being used to date (such as COM, Active Server pages (ASP), etc.)**
  - ⇒ **Bringing an opponent to Sun's Java technology**
  - ⇒ **Better coverage of mobile devices**
  - ⇒ **Simplified deployment of applications (fighting so called *DLL hell*)**
  - ⇒ **Better response to security issues**

# Managed code

- ◆ The major difference between .NET and previous object models is the use of **managed code**
  - ⇒ not executed by a processor in hardware, but by a virtual processor emulated by a virtual machine
- ◆ The code resides in so-called **assemblies** which resemble DLLs but are in addition equipped with **metadata** describing their identity, version number, content, and other things.
- ◆ The **virtual machine** provides a type system which permits data and classes to be shared across software written in a variety of several programming languages

# .NET Framework architecture



# What is new ?

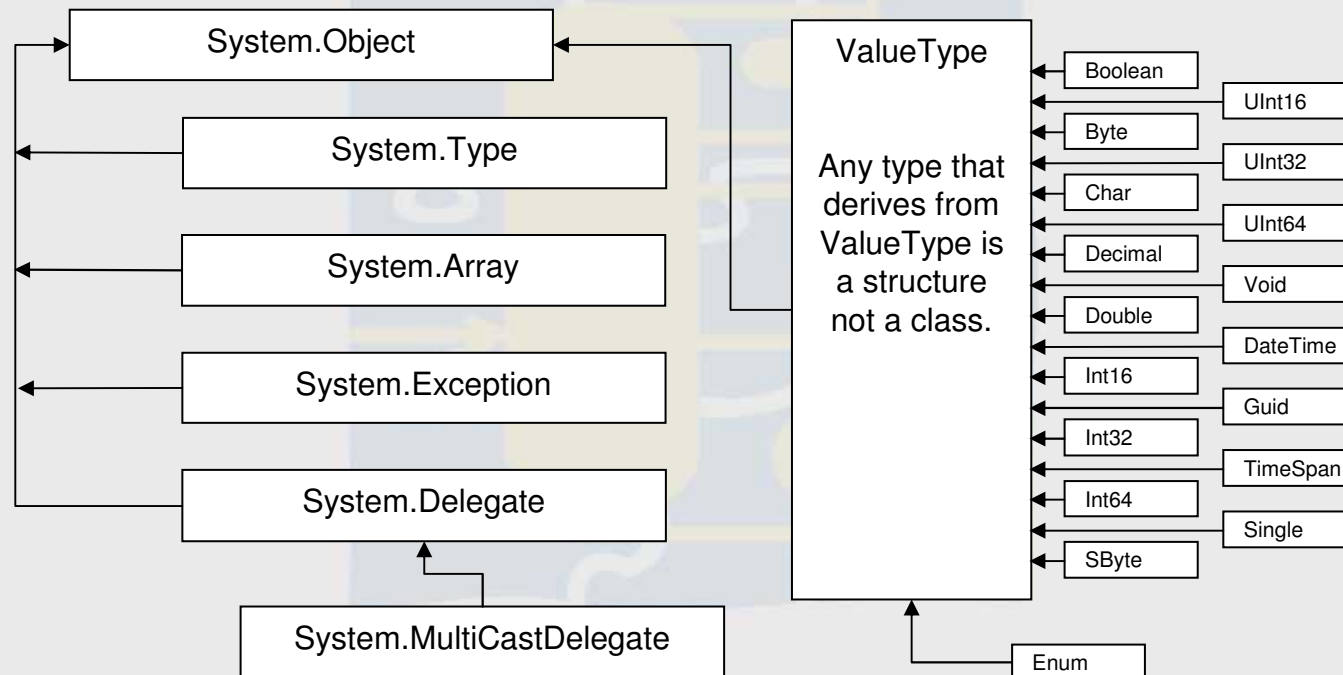
- ◆ **An open concept: specification accepted by ECMA**  
(European Computer Manufacturer's Association)
  - ⇒ Implementations for other platforms available
  - ⇒ Same code can run on different platforms
    - **dotGNU (Unix)**
    - **Mono (Unix)**
    - **.NET compact framework (mobile devices)**
- ◆ **xcopy deployment, no registration needed**
- ◆ **Language interoperability**
  - ⇒ One execution framework for various languages
  - ⇒ Seamless mixing and matching
- ◆ **Security, reflection, metadata**



# Languages of the .NET framework

- ◆ All languages are compiled into MSIL (MS Intermediate Language) at development time
- ◆ MSIL is compiled into real machine code at run time
- ◆ Many languages are available
  - ⇒ **C#, Visual Basic, C++, Python, ...**
- ◆ Just different representations of underlying MSIL
- ◆ Common language specification provides binding elements
  - ⇒ **Common type system**

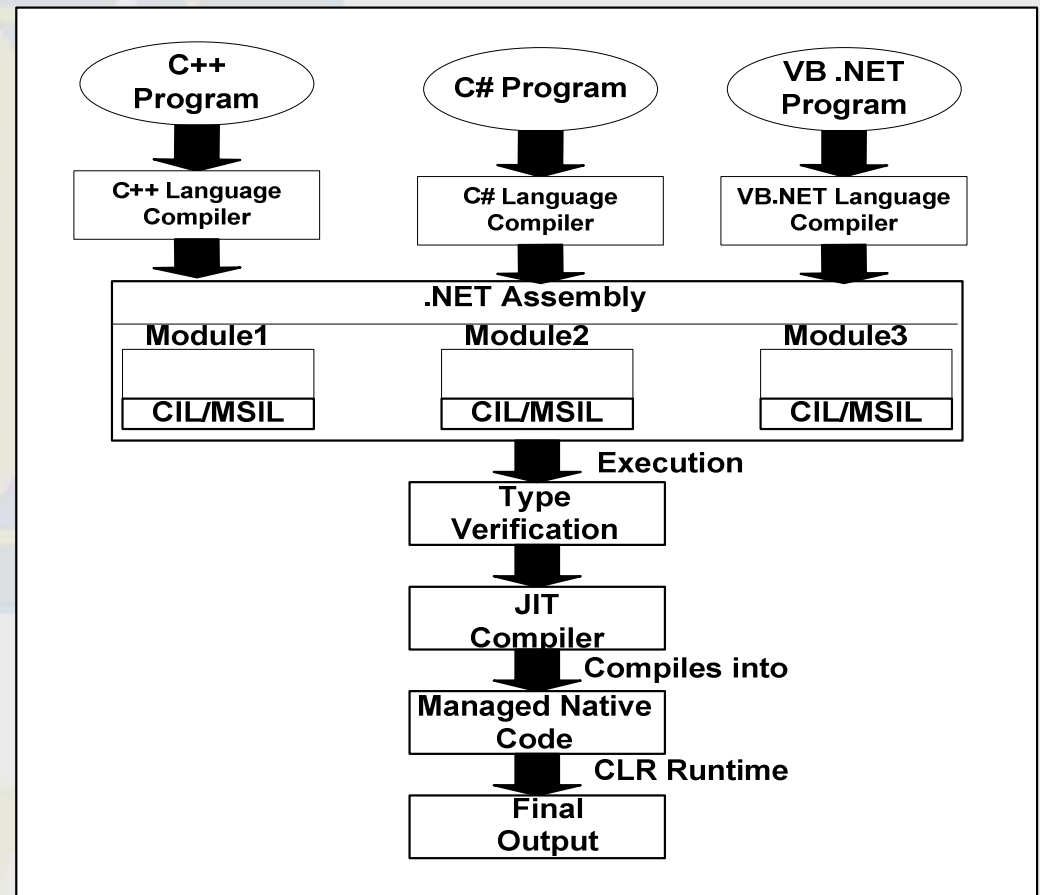
# Types in the common type system



- ◆ Embedded in .NET framework
- ◆ No need for explicit type conversions across programming languages

# Language interoperability in .NET

- ◆ Really transparent development across languages
- ◆ Interoperability is achieved in the MSIL layer
- ◆ Even inheritance is easily achieved across language boundaries



# Outline

- ◆ The role of middleware in CAPE-OPEN
- ◆ Why .NET, and what it is
- ◆ **.NET/COM CAPE-OPEN Interoperability**
- ◆ CO-LaN's roadmap for .NET

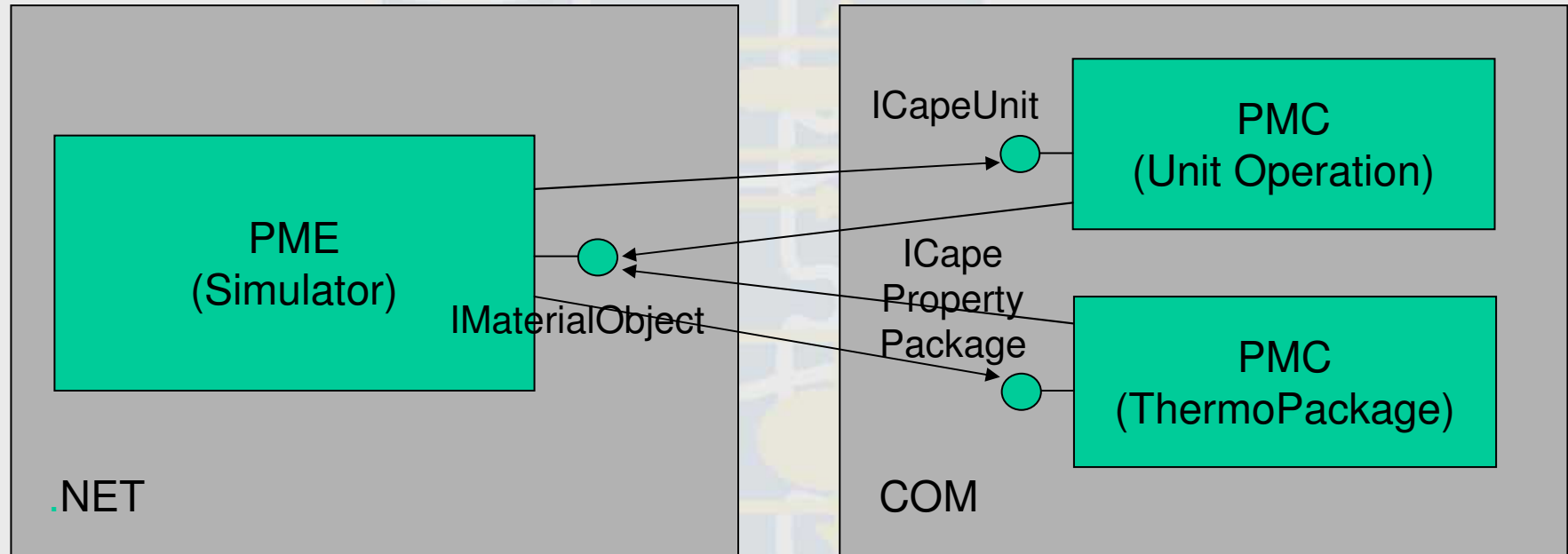


# COM interoperability

- ◆ Large effort made in .NET to continue use of COM
  - ⇒ Slow migration path essential to .NET success
  - ⇒ e.g. MS Office (Automation, Add-Ins), Explorer, ...
- ◆ Well-known approach for interoperability:
  - ⇒ Write a wrapper (bridge)
  - ⇒ Cumbersome and difficult
  - ⇒ .NET: wrappers are generated automatically !
- ◆ COM/.NET Wrappers are called Interop assemblies
  - ⇒ Intermediate pieces of software bridging COM / .NET



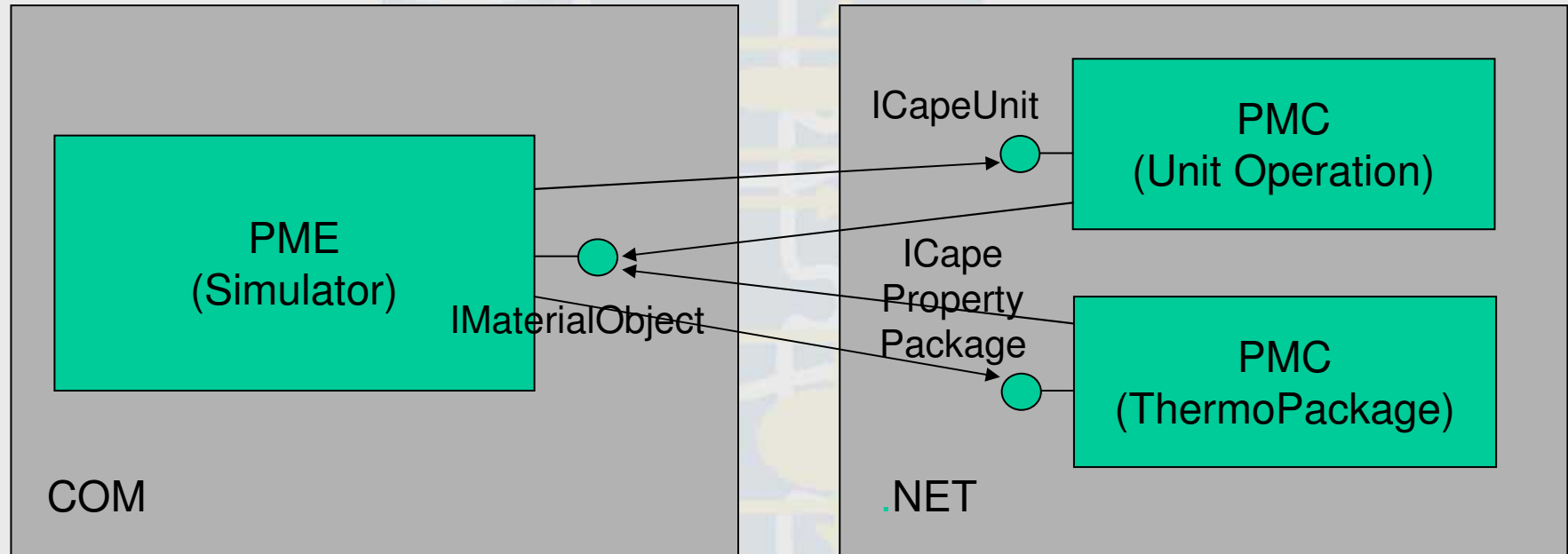
# Interoperability scenarios in CAPE-OPEN



.NET based PME, COM-based PMCs

- ◆ **Material object:** use .NET object in COM component
- ◆ **ICapeUnit, ICapeThermoPropertySystem, ...:** use COM component in .NET

# Interoperability scenarios in CAPE-OPEN



COM based PME, .NET-based PMCs

- ◆ **Material object: use COM object in .NET**
- ◆ **ICapeUnit, ICapeThermoPropertySystem, ...: provide .NET interfaces to COM**

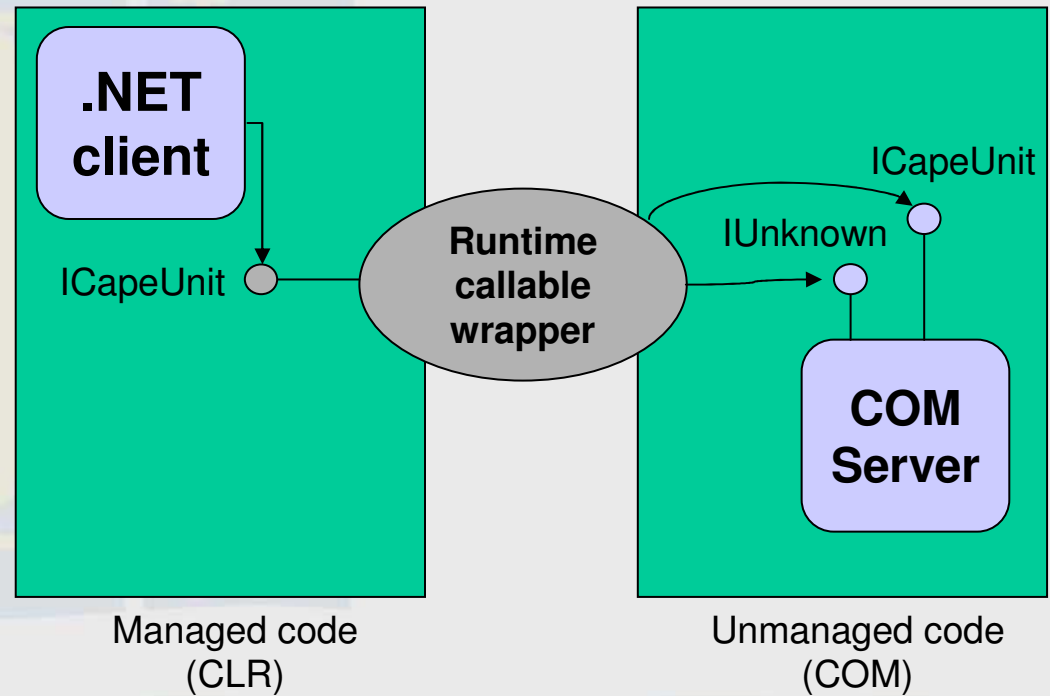
# Using COM components in .NET

Runtime callable wrapper

- ◆ Provide interfaces implemented by COM component
- ◆ Forward method invocations
- ◆ Translate types
  - ⇒ String
  - ⇒ Arrays
- ◆ Reference counting
- ◆ Generated e.g. from .tlb by *tlbimp.exe*

Use cases

- ◆ Existing CAPE-OPEN components in a new context (PME)
- ◆ Material object from COM-based PME in .NET-based PMC



# Responsibilities of RCWs

- ◆ **Mapping types**
  - ⊗ Simple types: int, bool, ...
  - ⊗ Strings: various flavors to simple .NET strings
  - ⊗ Arrays: Safearray, ... to simple .NET arrays
  - ⊗ References (IUnknown) to System.Object
- ◆ **Mapping system interfaces**
  - ⊗ IUnknown: reference counting sync'ed with .NET object lifecycle
  - ⊗ IDispatch: accessible via reflection
  - ⊗ IConnectionPointContainer: delegates in .NET (callback)
  - ⊗ IErrorInfo: forwarded as exception
  - ⊗ IEnumVariant: implemented as IEnumerable in .NET
- ◆ **Mapping proprietary interfaces**
  - ⊗ Provide native .NET interfaces for „proprietary“ interfaces
  - ⊗ Properties mapped to .NET properties
  - ⊗ Methods mapped, parameters accordingly



# Outline

- ◆ The role of middleware in CAPE-OPEN
- ◆ Why .NET, and what it is
- ◆ .NET/COM CAPE-OPEN Interoperability
- ◆ **CO-LaN's roadmap for .NET**



# Possible applications of CO with .Net

- ◆ **COM PMC in a COM PME**
  - ⇒ a.k.a. Classic CAPE-OPEN
- ◆ **.NET PMC in a COM PME**
  - ⇒ Typical scenario
  - ⇒ Example develop a unit op for use in existing PME
- ◆ **COM PMC in .NET PME**
  - ⇒ Enable your PME to use legacy PMCs
- ◆ **.NET PMC in .NET PME**
  - ⇒ Next generation modeling paradigm.
  - ⇒ Prepare your PMCs for the future.



# .NET interoperability with COM

- ◆ **CO-Lan created a document entitled “.NET Interoperability Guidelines”, which is available through the CO-LaN website.**
- ◆ **This document is based on the experience gained from**
  - ⇒ **The development of a .NET based PME at US EPA**
  - ⇒ **The migration of unit operation and thermodynamic models into .NET-enabled CO-compliant PMCs at AixCAPE.**
  - ⇒ **Additional input from the capeopentoolkit.net developed at University of Trieste by Pr. Fermeglia’s MOSE group**
- ◆ **Interoperability based upon .NET is a workable solution for the period of time in which COM and .NET implementations will exist besides each other.**



# The Roadmap

- ◆ **Create Primary Interop Assemblies from existing COM Type Libraries**
- ◆ **New interfaces will include .NET-based definitions.**
- ◆ **CAPE-OPEN will become .NET-based. Future versions will consist of .NET assemblies.**
  - ⇒ **COM interface definitions will be exported from .NET assemblies.**
- ◆ **.NET-based interface class assemblies will be formally published on colan.org.**



# CO-LaN Plans for 2007

- ◆ A .NET CO course was given during the annual European CAPE-OPEN conference in Heidelberg, last March
  - ⇒ 20 attendees for one full day
- ◆ In addition, CO-Lan plans to deliver:
  1. A guidance document that can be used by CO-LaN SIGs to develop/update to native .Net-based interfaces.
  2. Prototype .Net unit operation or property package as deemed appropriate.
    - The prototype source code will be made available to end-users wanting to develop .Net components.
  3. A wiki-based website for sharing problems and solutions for all SIGs



# CO-LaN provides a roadmap for the evolution of CAPE-OPEN into the .NET environment.

William M. Barrett, Michel Pons, Lars von Wedel, and Bertrand Braunschweig

